Online Unsupervised Neural-Gas Learning Method for Infinite Data Streams

Mohamed-Rafik Bouguelia, Yolande Belaïd, Abdel Belaïd

Online Unsupervised Neural-gas Learning Method for Infinite Data Streams

Mohamed-Rafik Bouguelia, Yolande Belaïd, and Abdel Belaïd

Université de Lorraine - LORIA,
UMR 7503,Vandoeuvre-les-Nancy, F-**54506**, France
{mohamed.bouguelia, yolande.belaid, abdel.belaid}@loria.fr
http://www.loria.fr

Abstract. We propose an unsupervised online learning method based on the "growing neural gas" algorithm (GNG), for a data-stream configuration where each incoming data is visited only once and used to incrementally update the learned model as soon as it is available. The method maintains a model as a dynamically evolving graph topology of data-representatives that we call neurons. Unlike usual incremental learning methods, it avoids the sensitivity to initialization parameters by using an adaptive parameter-free distance threshold to produce new neurons. Moreover, the proposed method performs a merging process which uses a distance-based probabilistic criterion to eventually merge neurons. This allows the algorithm to preserve a good computational efficiency over infinite time. Experiments on different real datasets, show that the proposed method is competitive with existing algorithms of the same family, while being independent of sensitive parameters and being able to maintain fewer neurons, which makes it convenient for learning from infinite data-streams.

Keywords: Incremental Learning:Unsupervised Neural Learning:Online Learning:Data Streams

1 INTRODUCTION

Recently, research focused on designing efficient algorithms for learning from continuously arriving streams of data, in an incremental way, where each data can be visited only once and processed dynamically as soon as it is available. Particularly, unsupervised incremental neural learning methods take into account relations of neighbourhood between representatives, and show a good clustering performance. Among these methods, GNG algorithm [4] has attracted considerable attention. It allows dynamic creation and removal of neurons (representatives) and edges between them during learning by maintaining a graph topology using a competitive Hebbian Learning strategy [12]. Each edge has an associated age which is used in order to remove old edges and keeps the topology dynamically updated. After adapting the graph topology using a fixed number of data-points from the input space (i.e. a time period), a new neuron is inserted

between the two neighbouring neurons that cumulated the most important error. Unlike usual clustering methods (e.g. Kmeans), it does not require initial conditions such as a predefined number of representatives and their initialization. This represents an important feature in the context of data streams where we have no prior knowledge about the whole dataset. However, in GNG, the creation of a new neuron is made periodically, and a major disadvantage concerns the choice of this period. For this purpose, some adaptations that relaxes this periodical evolution have been proposed. The main incremental variants are IGNG [5], I2GNG [6] and SOINN [7]. Unfortunately, the fact that these methods depend on some sensitive parameters that must be specified prior to the learning, reduces the importance of their incremental nature. Moreover, large classes are unnecessarily modelled by many neurons representing many small fragments, and leading to a significant drop of computational efficiency over time.

In this paper we propose a GNG based incremental learning algorithm (AING) where the decision of producing a new neuron from a new coming data-point is based on an adaptive parameter-free distance threshold. The algorithm overcomes the shortcoming of excessive number of neurons by condensing them based on a probabilistic criterion, and building a new topology with a fewer number of neurons, thus preserving time and memory resources. The algorithm depends only on a parameter generated by the system requirements (e.g. allowed memory budget), and unlike the other algorithms, no parameter related to a specific characteristics dataset needs to be specified. Indeed, it can be really difficult for a user to estimate all the parameters that are required by a learning algorithm. According to [13], "A parameter-free algorithm would limit our ability to impose our prejudices, expectations, and presumptions on the problem at hand, and would let the data itself speak to us". An algorithm which uses as few parameters as possible without requiring prior knowledge is strongly preferred, especially when the whole dataset is not available beforehand (i.e. a data-stream configuration).

This paper is organized as follows. In section 2, we describe a brief review of some incremental learning methods (mainly the GNG based ones), and analyse their problems. Then the algorithm we propose is presented in section 3. In section 4, we expose our experimental evaluation on synthetic and real datasets. In section 5, we give the conclusion and present some perspectives of this work.

2 RELATED WORK

Before describing some incremental methods and discussing their related problems, we firstly give some notations to be used in the rest of this paper: x refers to a data-point, y to a neuron, X_y is the set of data-points that are already assigned to neuron y, V_y is the set of current neurons that are neighbours of y (neurons linked to y by an edge), w_y is the reference vector of neuron y, and $n_y = |X_y|$ is the number of data-points currently assigned to y.

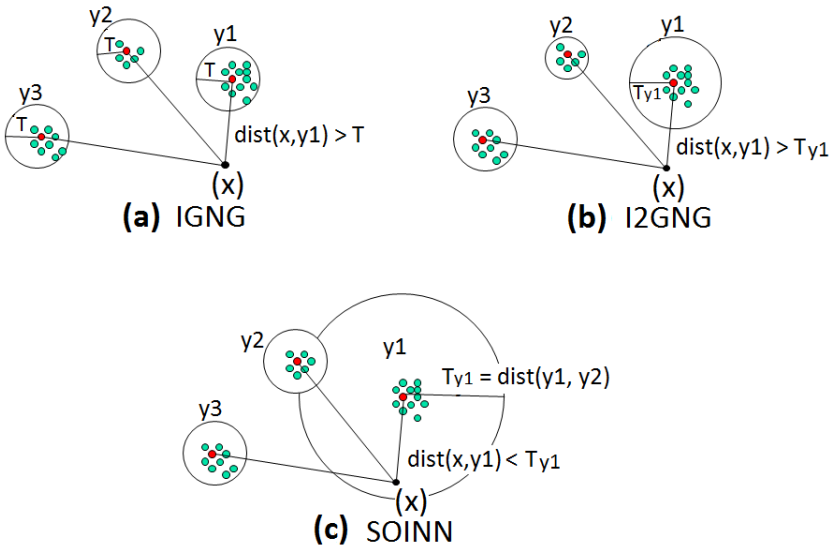

Fig. 1. Threshold based methods

The basic idea of the Incremental Growing Neural Gas algorithm (IGNG) [5] is that the decision of whether a new coming data-point x is close enough to its nearest neurons is made according to a fixed distance threshold value T (Figure 1(a)). Nevertheless, the main drawback of this approach is that the threshold T is globally the same for all neurons and must be provided as a parameter prior to the learning. There is no way to know beforehand which value is convenient for T, especially in a configuration where the whole dataset is not available.

I2GNG [6] is an improved version of IGNG where each neuron y has its own local threshold value (Figure 1(b)) which is continuously adapted during learning. If there is currently no data-point assigned to a neuron y, then its associated threshold is a default value T which is an input parameter given manually as in IGNG; otherwise, the threshold is defined as $\bar{d}+\alpha\sigma$, where $\bar{d}$ is the mean distance of y to its currently assigned data-points, σ is the corresponding standard deviation, and α a parameter. Choosing "good" values for parameters T and α is important since the evolution of the threshold will strongly depends on them. For instance, if they are set at a relatively small value (depending on the dataset) then many unnecessary neurons are created. On the other hand, if their values are relatively big, then some data-points may wrongly be assigned to some close clusters. This clearly makes systems using such an algorithm dependent on an expert user and gives less emphasis to its incremental nature.

In the Self-Organizing Incremental Neural Network (SOINN) [7], the threshold of a given neuron y is defined as the maximum distance of neuron y to its current neighbours if they exist, otherwise it is the distance of y to its nearest neuron among the existing ones (Figure 1(c)). SOINN's threshold is often more

sensitive to the creation order of neurons (induced by the arrival order of data-points), especially in first steps. Furthermore, SOINN deletes isolated neurons and neurons having only one neighbour when the number of input data-points is a multiple of a parameter λ (a period).

Many other parameter-driven methods have been designed especially for data stream clustering, among this methods we can cite: Stream [1], CluStream [2] and Density-Based clustering for data stream [3].

There are several variants of Kmeans that are said "incremental". The one proposed in [8] is based on a creation cost of cluster centers; the higher it is, the fewer is the number of created clusters. The cost is eventually incremented and the cluster centers are re-evaluated. However, the algorithm assumes that the size of the processed dataset is known and finite.

3 PROPOSED ALGORITHM (AING)

In this section, we propose a scalable unsupervised incremental learning algorithm that is independent of sensitive parameters, and dynamically creates neurons and edges between them as data come. It is called "AING" for Adaptive Incremental Neural Gas.

3.1 General behaviour

The general schema of AING can be expressed according to the following three cases. Let y_1 and y_2 respectively be the nearest and the second nearest neurons from a new data-point x, such that $\text{dist}(y_1, x) < \text{dist}(y_2, x)$:

1. if x is *far enough* from y_1: a new neuron y_{new} is created at x (see Figure 2, 1^{st} case).
2. if x is *close enough* to y_1 but *far enough* from y_2: a new neuron y_{new} is created at x, and linked to y_1 by a new edge (see Figure 2, 2^{nd} case).
3. if x is *close enough* to y_1 and *close enough* to y_2 (see Figure 2, 3^{rd} case):
   - move y_1 and its neighbouring neurons towards x, i.e. modify their reference vectors to be less distant from x.
   - increase the age of y_1's edges
   - link y_1 to y_2 by a new edge (reset its age to 0 if it already exists)
   - activate the neighbouring neurons of y_1
   - delete the old edges if any

An age in this context is simply a value associated to each existing edge. Each time a data-point x is assigned to the winning neuron y_1 (the 3^{rd} case), the age of edges emanating from this neuron is increased. Each time a data-point x is close enough to neurons y_1 and y_2, the age of the edge linking this two neurons is reset to 0. If the age of an edge continues to increase without being reset, it will reaches a maximum age value and the edge will be considered "old" and thus removed.

A data-point x is considered *far* (respectively *close*) enough from a neuron y, if the distance between x and y is higher (respectively smaller) than a threshold T_y. The following subsection shows how this threshold is defined.

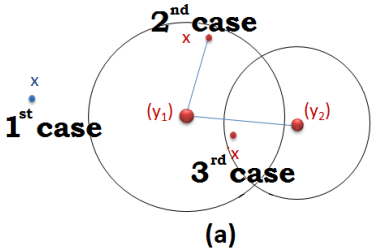

Fig. 2. AING general cases

3.2 AING distance threshold

Since the input data distribution is unknown, we define a parameter-free adaptive threshold T_y which is local to each neuron. The idea is to make the threshold T_y of a neuron y, dependent on the distances to data in its neighbourhood. The neighbourhood of y consists of data-points previously assigned to y (for which y is the nearest neuron), and data-points assigned to the neighbouring neurons of y (neurons that are linked to y by an edge).

According to formula 1, the threshold T_y of a neuron y is defined as the sum of distances from y to its data-points, plus the sum of weighted distances from y to its neighbouring neurons[1], averaged on the total number of the considered distances. In the case where the neuron y has no data-points that were already assigned to it (X_y is empty) and has no neighbour (V_y is empty), then we consider the threshold T_y as the half distance from y to its nearest neuron.

$$T_y = \begin{cases} \frac{\sum_{e \in X_y} \text{dist}(y,e) + \sum_{e \in V_y} |X_e| \times \text{dist}(y,e)}{|X_y| + \sum_{e \in V_y} |X_e|} & \text{if } X_y \neq \emptyset \vee V_y \neq \emptyset \\ \frac{\text{dist}(y,\tilde{y})}{2}, \tilde{y} = \underset{\tilde{y} \neq y}{\text{argmin}}\, \text{dist}(y,\tilde{y}) & \text{otherwise} \end{cases} \tag{1}$$

Note that we do not need to save data-points that are already seen in order to compute this threshold. It is incrementally computed each time a new data-point comes, by updating some information associated to each neuron (e.g. number of data-points associated to a neuron, the sum of their distances to this neuron, etc.). If we consider the example of Figure 3, there are 3 data-points assigned to y_1 (namely x_1, x_2 and x_3), and two neurons that are neighbours of y_1 (namely y_2 with 4 assigned data-points, and y_3 with 5 data-points). In this case, the threshold associated to the neuron y_1 is computed as

$$T_{y_1} = \frac{\text{dist}(y_1,x_1) + \text{dist}(y_1,x_2) + \text{dist}(y_1,x_3) + 4\,\text{dist}(y_1,y_2) + 5\,\text{dist}(y_1,y_3)}{3+4+5}$$

As we can see, the proposed threshold is independent of parameters and evolves dynamically according to the data and the topology of neurons.

[1] The distance is weighted by the number of data-points associated to the neighbouring neuron

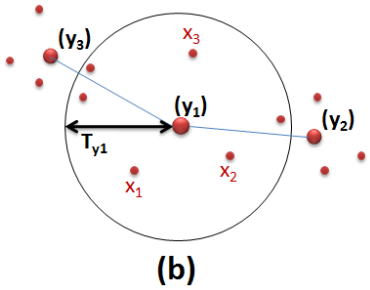

Fig. 3. AING threshold definition

3.3 AING merging process

Since data is processed online, it is usually common that algorithms for data stream clustering generate many cluster representatives. However, this may significantly compromise the computational efficiency over time. Instead of introducing parameters in the threshold computation to control the number of created neurons, AING can eventually reduce the number of neurons through the merging process. Indeed, when the number of current neurons reaches an upper bound (*up_bound*), some close neurons can be merged.

The merging process globally follows the same scheme as previously, but instead of relying on a hard rule based on a threshold, it uses a more relaxed rule based on a probabilistic criterion. Saying that "a neuron y is *far enough* from its nearest neuron $\tilde{y}$" is expressed as the probability that y will not be assigned to $\tilde{y}$, according to the formula $P_{y,\tilde{y}} = \frac{|X_y| \times \text{dist}(y,\tilde{y})}{\kappa}$. This probability is proportional to the distance between the two neurons ($\text{dist}(y,\tilde{y})$) and to the number of data-points assigned to y ($|X_y|$), that is, the more y is large and far from $\tilde{y}$, the more likely it is to remain not merged. The probability is in contrast inversely proportional to a variable κ, which means that by incrementing κ, any given neuron y will have more chance to be merged with its nearest neuron. Let $\bar{d}$ be the mean distance of all existing neurons to the center-of-mass of the observed data-points. κ is incremented by $\kappa = \kappa + \bar{d}$ each time the neurons need to be more condensed, i.e. until the merging process takes effect and the number of neurons becomes less than the specified limit *up_bound*. Note that $P_{y,\tilde{y}}$ as specified may be higher than 1 when κ is not yet sufficiently big; a better formulation would be $P_{y,\tilde{y}} = \min(\frac{|X_y| \times \text{dist}(y,\tilde{y})}{\kappa}, 1)$, to guarantee it to be always a true probability.

The merging process is optional. Indeed, *up_bound* can be set to $+\infty$ if desired. Alternatively, the merging process can be triggered at any time chosen by the user, or by choosing the parameter *up_bound* according to some system requirements such as the memory budget that we want to allocate for the learning task, or the maximum latency time tolerated by the system due to a high number of neurons.

Finally, the code is explicitly presented in Algorithms 1 and 2, which provide an overall insight on the AING method. They both follow the same scheme described in section 3.1. Algorithm 1 starts from scratch and incrementally processes each data-point from the stream using the adaptive distance threshold described in section 3.2. When the number of current neurons reaches a limit, Algorithm 2 is called and some neurons are grouped together using the probabilistic criterion described in section 3.3. We just need to point out two additional details appearing in our algorithms:

- If a data-point x is close enough to its two nearest neurons y_1 and y_2, it is assigned to y_1 and the reference vector of this later and its neighbours are updated (i.e. they move towards x) by a learning rate: ϵ_b for y_1 and ϵ_n for its neighbours (lines 15-17 of Algorithm 1). Generally, a too big learning rate implies instability of neurons, while a too small learning rate implies that neurons do not learn enough from their assigned data. Typical values are $0 < \epsilon_b \ll 1$ and $0 < \epsilon_n \ll \epsilon_b$. In AING, $\epsilon_b = \frac{1}{|X_{y_1}|}$ is slowly decreasing proportionally to the number of data-points associated to y_1, i.e. the more y_1 learns, the more it becomes stable, and ϵ_n is simply heuristically set to 100 times smaller than the actual value of ϵ_b (i.e. $\epsilon_n \ll \epsilon_b$)
- Each time a data-point is assigned to a winning neuron y_1, the age of edges emanating from this neuron is increased (line 14 of Algorithm 1). Let n_{max} the maximum number of data-points assigned to a neuron. A given edge is then considered "old" and thus removed (line 19 of Algorithm 1) if its age becomes higher than n_{max}. Note that this is not an externally-set parameter, it is the current maximum number of data-points assigned to a neuron among the existing ones.

4 EXPERIMENTAL EVALUATION

4.1 Experiments on synthetic data

In order to test AING's behaviour, we perform an experiment on artificial 2D data of 5 classes (Figure 4(a)) composed of a Gaussian cloud, a uniform distribution following different shapes, and some uniformly distributed random noise. Figure 4(b) and 4(c) show the topology of neurons obtained without using the merging process ($up\_bound = +\infty$), whereas for Figure 4(d) and 4(e), the merging process was also considered. However, for Figure 4(b) and 4(d), the data were given to AING class by class in order to test the incremental behaviour of AING. The results show that AING perfectly learns the topology of data and confirms that it has good memory properties. On the other hand, for Figure 4(c) and 4(e) the arrival order of data was random. The results show that AING performs well, even if the arrival order of data is random.

Algorithm 1 AING Algorithm (up_bound)

init graph G with the two first coming data-points
$\kappa = 0$
while some data-points remain unread **do**
    get next data-point x, update $\bar{d}$ accordingly
    let y_1, y_2 the two nearest neurons from x in G
    get T_{y_1} and T_{y_2} according to formula 1
    if $dist(x, w_{y_1}) > T_{y_1}$ **then**
        $G \leftarrow G \cup \{y_{new}/w_{y_{new}} = x\}$
    else
        if $dist(x, w_{y_2}) > T_{y_2}$ **then**
            $G \leftarrow G \cup \{y_{new}/w_{y_{new}} = x\}$
            connect y_{new} to y_1 by an edge of age 0
        else
            increase the age of edges emanating from y_1
            let $\epsilon_b = \frac{1}{|X_{y_1}|}, \epsilon_n = \frac{1}{100 \times |X_{y_1}|}$
            $w_{y_1} += \epsilon_b \times (x - w_{y_1})$
            $w_{y_n} += \epsilon_n \times (x - w_{y_n}), \forall y_n \in V_{y_1}$
            connect y_1 to y_2 by an edge of age 0
            remove old edges from G if any
        end if
    end if
    while number of neurons in $G >$ up_bound **do**
        $\kappa = \kappa + \bar{d}$
        $G \leftarrow$ Merging(κ, G)
    end while
end while

4.2 EXPERIMENTS ON REAL DATASETS

We consider in our experimental evaluation, AING with and without the merging process[2], some main incremental neural clustering algorithms, and an accurate incremental Kmeans [8] as a reference in comparing the results.

We consider a total of six datasets of different size and dimensions. Three standard public handwritten digit datasets (i.e. Pendigit and Optdigit from the UCI repository [9], and Mnist dataset [10]), and three different datasets of documents represented as bag of words, taken from a real administrative documents processing chain:

- Pendigit: 7494 data for learning, 3498 data for testing, 17 dimensions, 10 classes.
- Optdigit: 3823 data for learning, 1797 for testing, 65 dimensions, 10 classes.
- Mnist: 60000 data for learning, 10000 for testing, 784 dimensions, 10 classes.
- 1^{st} documentary dataset: 1554 data for learning, 777 for testing, 272 dimensions, 143 classes.

[2] We will refer to AING without the merging process by AING1, and to AING with the merging process by AING2

Algorithm 2 Merging (κ, G)

init $\tilde{G}$ with two neurons chosen randomly from G
for all $y \in G$ **do**
  let $\tilde{y}_1$, $\tilde{y}_2$ the two nearest neurons from y in $\tilde{G}$
  let $d_1 = dist(w_y, w_{\tilde{y}_1}), d_2 = dist(w_y, w_{\tilde{y}_2})$
  if $\underset{uniform}{\text{random}}([0,1]) < \min(\frac{n_y \times d_1}{\kappa}, 1)$ **then**
    $\tilde{G} \leftarrow \tilde{G} \cup \{\tilde{y}_{new} / w_{\tilde{y}_{new}} = w_y\}$
  else
    if $\underset{uniform}{\text{random}}([0,1]) < \min(\frac{n_y \times d_2}{\kappa}, 1)$ **then**
      $\tilde{G} \leftarrow \tilde{G} \cup \{\tilde{y}_{new} / w_{\tilde{y}_{new}} = w_y\}$
      connect $\tilde{y}_{new}$ to $\tilde{y}_1$ by an edge of age 0
    else
      increase age's edges emanating from $\tilde{y}_1$
      Let $\epsilon_b = \frac{1}{|X_{\tilde{y}_1}|}, \epsilon_n = \frac{1}{100 \times |X_{\tilde{y}_1}|}$
      $w_{\tilde{y}_1} += \epsilon_b \times (w_y - w_{\tilde{y}_1})$
      $w_{\tilde{y}_n} += \epsilon_n \times (w_y - w_{\tilde{y}_n}), \forall \tilde{y}_n \in V_{\tilde{y}_1}$
      connect $\tilde{y}_1$ to $\tilde{y}_2$ by an edge of age 0
      remove old edges from $\tilde{G}$ if any
    end if
  end if
end for
return $\tilde{G}$

- 2^{nd} documentary dataset. 2630 data for learning, 1315 for testing, 278 dimensions, 24 classes.
- 3^{rd} documentary dataset. 3564 data for learning, 1780 for testing, 293 dimensions, 25 classes.

In addition to the number of produced representatives and the number of required parameters, we consider as evaluation measures the recognition rate (R) and the v-measure (V) [11]. Basically, v-measure is an entropy-based measure which expresses the compromise between homogeneity and completeness of the produced clusters and gives an idea about the ability to generalize to future data. Indeed, according to [11], it is important that clusters contain only data-points which are members of a single class (perfect homogeneity), but it is also important that all the data-points that are members of a given class are elements of the same cluster (perfect completeness).

For each algorithm, we repeat many experiments by slightly varying the parameter values needed by each of them. We finally keep the parameter values matching the best clustering results according to the considered evaluation measures.

The results obtained on the 3 first datasets are shown in Table 1, where AING1 (respectively AING2) refers to AING without (respectively with) the merging process. From Table 1, we see that concerning the 1^{st} dataset, Kmeans achieves a better v-measure, and maintains fewer representatives, but does not

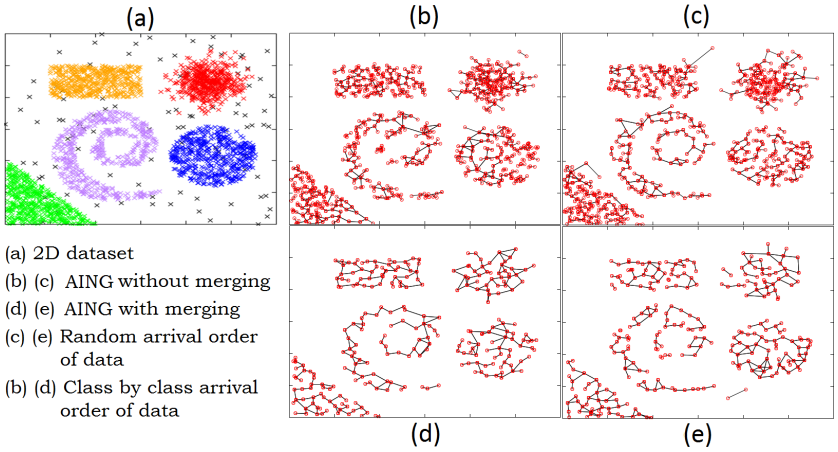

Fig. 4. The built topology of activated neurons, with and without the merging process

reach a recognition rate which is comparable to the other algorithms. Although AING1 (without the merging process) is independent of external parameters, it realises almost the same recognition rate and v-measure as SOINN and I2GNG. AING2 (with the merging process) produces fewer neurons and the recognition rate as well as the v-measure are improved further. Concerning the 2^{nd} dataset (Optdigit), AING1 realises the greatest performances. With AING2, the number of neurons is considerably reduced and a better compromise between homogeneity and completeness is achieved. The recognition rate is a little worse than the AING1, but still very close to the highest rate obtained by the other algorithms. Concerning the Mnist dataset, AING2 achieved the best performances.

Table 2 shows the results obtained on the documentary datasets. Roughly, we can make the same conclusions as with the previous datasets. AING1 performs well, although it does not require other pre-defined parameters. However, when using the merging process (AING2) on these datasets, the obtained results are of lower quality than those obtained with AING1. This is due to the fact that these documentary datasets are not very large and that the obtained neurons are not sufficient to represent well all the different classes. Indeed, the documentary datasets contains much more classes than the 3 first handwritten digits datasets. The merging process is thus more convenient when dealing with large datasets.

Figure 5 shows how the recognition rate changes with changing values of the upper bound parameter (up_bound) for some datasets. Due to the reason cited previously, for the documentary datasets, the results are better as the value of the parameter up_bound is higher (which implies more neurons). However, if we take as an example the Pendigit dataset, we can observe that for all values greater than or equal to 600 (i.e. most reasonable values that up_bound can take), the recognition rate is in $[97, 98]$ (i.e. around the same value). Note that for two experiments with a fixed value of up_bound, the result may slightly be different since the merging process is probabilistic. Furthermore, the maximum

Table 1. Validation on public standard datasets (R = Recognition rate, V = V-Measure, Params = Number of parameters)

Method	**Neurons**	**R %**	**V %**	**Params**
Pendigit dataset				
AING1	1943	97.427	52.538	**0**
AING2	1403	**97.827**	53.624	1
Kmeans	**1172**	97.055	**54.907**	3
SOINN	1496	97.341	52.222	3
I2GNG	2215	97.541	52.445	4
Optdigit dataset				
Method	**Neurons**	**R %**	**V %**	**Params**
AING1	1371	**97.718**	54.991	**0**
AING2	**825**	97.440	**55.852**	1
Kmeans	1396	97.495	52.899	3
SOINN	1182	96.82	53.152	3
I2GNG	1595	97.161	53.555	4
Mnist dataset				
Method	**Neurons**	**R %**	**V %**	**Params**
AING1	3606	94.06	45.258	**0**
AING2	**2027**	**94.21**	**46.959**	1
Kmeans	2829	94.04	45.352	3
SOINN	2354	93.95	44.293	3
I2GNG	5525	94.10	43.391	4

number of neurons that can be generated for this example is 1943, thus, for values of up_bound in $[1943, +\infty[$, the merging process does not take place and AING2 performs exactly like AING1 (i.e. for AING on the Pendigit dataset $\forall$ up_bound $\in [1943, +\infty[$: R = 97.4271%).

Furthermore, the time required to incrementally integrate one data-point is strongly related to the current number of neurons (representatives) because the search for the nearest neurons from a new data-point is the most consuming operation. Figure 6 shows that AING is more convenient for a long-life learning task since it maintains a better processing time than the other algorithms over long periods of time learning, thanks to the merging process. The overall running time for the Mnist dataset (i.e. required for all the 60000 data-points) is 1.83 hours for AING, 2.57 hours for SOINN and 4.49 hours for I2GNG.

5 CONCLUSION AND FUTURE WORK

This paper presents an online unsupervised learning method which incrementally processes data from the data stream, without being sensitive to initialization parameters. It initially decides whether a new data-point should produce a new cluster representative by means of a parameter-free adaptive threshold associated to each existing representative, and evolving dynamically according to the data and the topology of neurons. Some representatives may eventually be assigned

Table 2. Validation on datasets of administrative documents (R = Recognition rate, V = V-Measure, Params = Number of parameters)

1^{st} documentary dataset				
Method	**Neurons**	**R %**	**V %**	**Params**
AING1	1030	**91.505**	**87.751**	**0**
AING2	1012	89.446	87.461	1
Kmeans	**1013**	90.862	86.565	3
SOINN	1045	88.545	87.375	3
I2GNG	1367	91.119	86.273	4
2^{nd} documentary dataset				
Method	**Neurons**	**R %**	**V %**	**Params**
AING1	1215	98.251	**57.173**	**0**
AING2	**1011**	97.490	**59.356**	1
Kmeans	1720	98.098	53.966	3
SOINN	1650	97.338	55.124	3
I2GNG	1846	**98.403**	54.782	4
3^{rd} documentary dataset				
Method	**Neurons**	**R %**	**V %**	**Params**
AING1	2279	**91.685**	60.922	**0**
AING2	**1897**	89.269	**62.367**	1
Kmeans	2027	91.179	60.192	3
SOINN	2437	88.707	61.048	3
I2GNG	2618	90.393	60.954	4

to others by means of a distance-based probabilistic criterion each time their number exceed a specified limit; thus, maintaining a better clusters completeness, and preserving time and memory resources.

Nonetheless, further work still needs to be done. One of our directions for future work is to provide some theoretical worst-case bounds on memory and time requirement, and allow the algorithm to automatically determine an appropriate upper bound for the number of representatives; this will allow AING to perform a long-life learning. Then, we want to integrate the algorithm in a case-based reasoning system for document analysis, whose case-base will be continuously maintained by the AING algorithm.

Another direction is the extension of this work to semi-supervised and active learning. Indeed, AING is unsupervised and can not be directly applied to any classification task. The work in [14] extends AING in order to be suitable for a text document classification task, by allowing the algorithm to learn from both labelled and unlabelled documents and to actively query (from a human annotator) the class-labels of only documents that are most informative for learning, thus saving annotation time and effort.

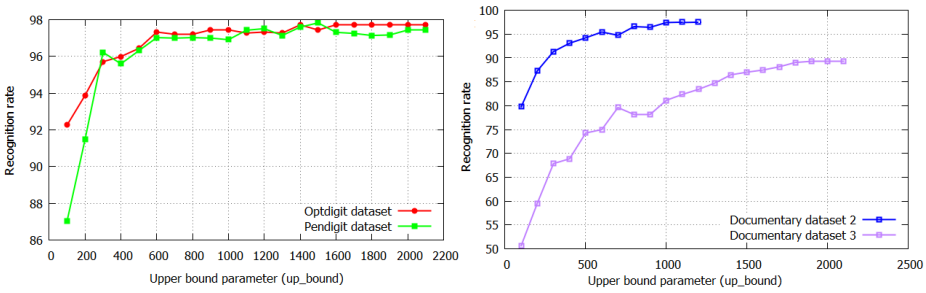

Fig. 5. The recognition rate achieved by AING according to the parameter up_bound for some datasets

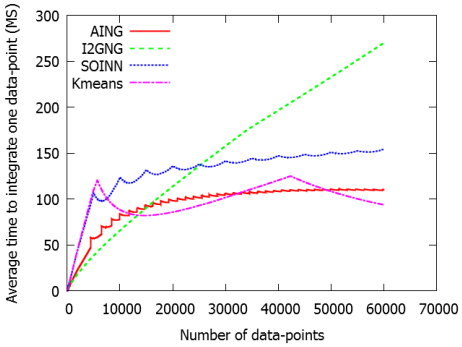

Fig. 6. The average time (in milliseconds) required to incrementally integrate one data-point (for the Mnist dataset)

References

1. Liadan O'Callaghan, Adam Meyerson, Rajeev Motwani, Nina Mishra, Sudipto Guha: Streaming-Data Algorithms for High-Quality Clustering. In: International Conference on Data Engineering, pp. 685–696. San Francisco, (2002)
2. Charu C Aggarwal, Jiawei Han, Jianyong Wang, Philip S Yu: A framework for clustering evolving data streams. In: Proceedings of the 29th international conference on Very large data bases, pp. 81–92. Berlin, Germany, (2003)
3. Y. Chen, L. Tu: Density-Based Clustering for Real-Time Stream Data. In: International Conference on Knowledge Discovery and Data Mining, pp. 133–142. (2007)
4. B. Fritzke: A growing neural gas network learns topologies. In: Neural Information Processing Systems, pp. 625–632. (1995)
5. Y. Prudent, A. Ennaji: An incremental growing neural gas learns topologies. In: International Joint Conference on Neural Networks, pp. 1211–1216. (2005)
6. H. Hamza, Y. Belaid, A. Belaid, B. Chaudhuri: Incremental classification of invoice documents. In: International Conference on Pattern Recognition, pp. 1–4. (2008)
7. F. Shen, T. Ogura, O. Hasegawa: An enhanced self-organizing incremental neural network for online unsupervised learning. Neural Networks, pp. 893–903. (2007)
8. M. Shindler, A. Wong, A. Meyerson: Fast and accurate k-means for large datasets. In: Neural Information Processing Systems, pp. 2375–2383. (2011)

9. A. Frank, A. Asuncion: The UCI machine learning repository, http://archive.ics.uci.edu/ml/. (2010)
10. LeCun Yann, Cortes Corinna: MNIST handwritten digit database, http://yann.lecun.com/exdb/mnist/. (2010)
11. A. Rosenberg, J. Hirschberg: V-measure: A conditional entropy-based external cluster evaluation measure. In: Neural Information Processing Systems, pp. 410–420. (2007)
12. T. Martinetz: Competitive hebbian learning rule forms perfectly topology preserving maps. In: Proceedings of the International Conference on Artificial Neural Networks, pp. 427–434. Amsterdam, Netherlands, (1993)
13. E. Keogh, S. Lonardi, C. A. Ratanamahatana: Towards parameter-free data mining. In: Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining, pp. 206–215. (2004)
14. M-R. Bouguelia, Y. Belaid, A. Belaid. A Stream-Based Semi-Supervised Active Learning Approach for Document Classification. To appear in: International Conference on Document Analysis and Recognition. (2013)